Conceptual Exploration with LLMs for Semantic Mapping in Lexical Typology

Vaibhav Singh

Technical University Dresden, Germany

Vaibhav.singh@mailbox,tu-dresden.de

Abstract

Semantic maps are fundamental tools in lexical typology for visualizing how languages organize meanings across conceptual space. Traditional semantic map construction relies on manual data collection through questionnaires with native speakers a labor-intensive process that becomes increasingly impractical with larger datasets. While concept lattices from Formal Concept Analysis (FCA) offer automated construction and enforce connectivity constraints through attribute implications, validating these implications across multiple languages remains challenging. This study introduces an LLM-assisted system that integrates large language models into the conceptual exploration framework to support systematic validation of semantic implications.

Keywords: Lexical Typology, Semantic Maps, Formal Concept Analysis, Concept Lattices, Large Language Models, Conceptual Exploration, Cross-linguistic Analysis

1 Introduction

Lexical typology is a subfield of linguistics that focuses on how words in different languages express related meanings and searches for patterns in how humans categorize reality through vocabulary [10][17]. The central question of the field is "*How do words in different languages cover a conceptual space of related meanings?"*, this question is generally asked with respect to a specific field and to answer the question lexical typologists employ a key analytical tool: the semantic map.

According to the Frame approach to lexical typology [16], there exists a universal set of minimal meanings (Frames) that are clustered differently across different languages. Semantic maps visualize how different languages organize these meanings across conceptual space and illustrate which combinations of meanings are admissible and which are forbidden.

1.1 Semantic map as graphs

A semantic map is an undirected graph where nodes represent individual meanings (Frames) and edges connect related meanings. These meanings and their connections are hypothesized to be universal across languages. Language-specific words are represented by mapping them onto connected subsets of this universal structure [10]. According to the semantic connectivity hypothesis [2], any word must correspond to a connected region within this conceptual space. In other words, if a word expresses two meanings, it must also express the intermediate meanings that connect them on the map. This reflects the principle that words group conceptually contiguous meanings rather than disconnected ones.

Complementing this, the **Economy Principle** [10] posits that *no edge is needed between nodes A and C if linguistic items expressing A and C always also express B*. This keeps semantic maps minimal by including only necessary connections to account for observed linguistic patterns.

Together, the Connectivity Hypothesis and the Economy Principle govern semantic map structure: the former defines which meanings can co-occur in a single word, while the latter minimizes the number of connections needed to explain cross-linguistic data.

Figure 1 and 2, shows an example of a semantic maps for the field "*Divide*", constructed from data in Table 1. The data is a fragment of the dataset published in [1]. Detailed explanation of the frames is given in the appendix A. The map in Figure 1 suggests that if the same word is used to describe the meanings "isolate constituent parts" and "distribute among others", then it can also be used to describe "separate into parts". However, it remains unclear whether such a word can also be used for describing "distribute among themselves". This is a problem with classical semantic maps: they cannot enforce such combinations without prohibiting other valid combinations. This is a well-known limitation of classical semantic maps: by accommodating all combinations of meanings attested in the data, they "predict much more than is actually found" [4].

Object	1. divide a single physical object into constituent parts	2. divide a set of elements into groups	3. mark the boundary	4. isolate constituent parts of a non-physical object	5. put into containers	6. cause to move or be apart	7. distribute among others	8. distribute among themselves
delit'	X	X	X	X	X		X	X
razbivat'		X	X	X				
razluchat'						X		
divide	X	X	X	X	X		X	X
share							X	X
split	X	X	X	X		X	X	X
separate		X	X	X	X	X		

Table 1. *Data from the semantic field 'divide'*

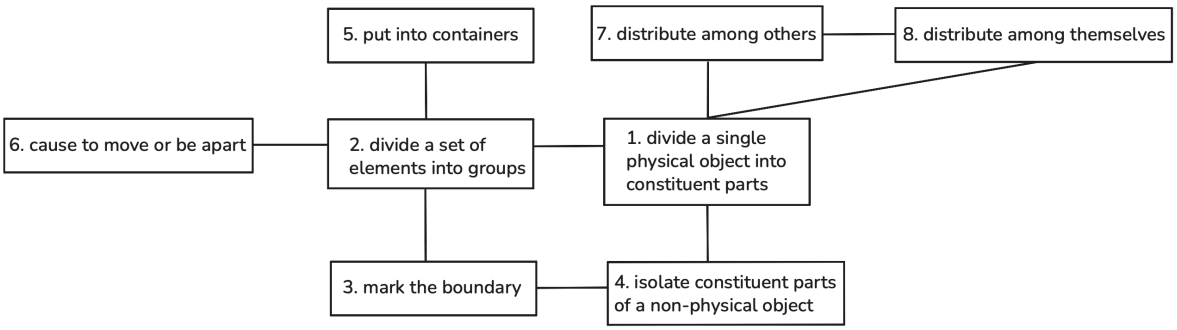
5. put into containers
7. distribute among others
8. distribute among themselves
6. cause to move or be apart
2. divide a set of elements into groups
1. divide a single physical object into constituent parts
3. mark the boundary
4. isolate constituent parts of a non-physical object

Figure 1: *Semantic map for the field "divide"*

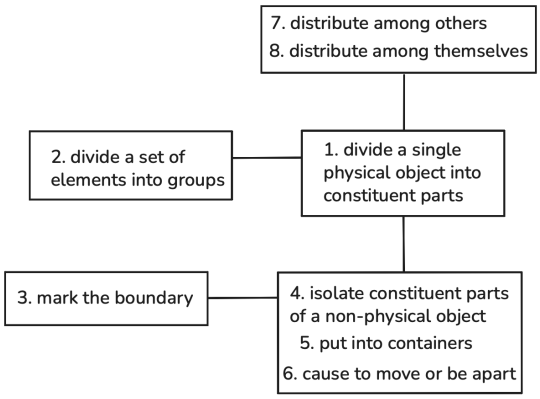
7. distribute among others
8. distribute among themselves
2. divide a set of elements into groups
1. divide a single physical object into constituent parts
3. mark the boundary
4. isolate constituent parts of a non-physical object
5. put into containers
6. cause to move or be apart

Figure 2: *Reduced semantic map for the field "divide"*

Another downside of semantic maps is that they are usually constructed manually, which becomes increasingly labor-intensive as dataset size grows. Alternate approaches to classical semantic maps have appeared like multidimensional scaling technique and concept lattices. MDS-based approaches [3][19] transform cross-linguistic similarity data into spatial representations, positioning meanings as points in a continuous multidimensional space where distances reflect semantic proximity. While MDS enables automatic sematic map construction, handles larger datasets and provides quantitative measures of similarity, it produces non-discrete visualizations that may be less intuitive for capturing categorical linguistic distinctions. Moreover, MDS maps do not inherently guarantee the connectivity constraint central to semantic map theory.

Concept lattices address both issues present in classical semantic maps: they can be constructed automatically from data and provide attribute implications that constrain attribute combinations. In the following sections we will discuss about concept lattices in detail.

1.2 Concept Lattice

The idea of concept lattices originates from a mathematical framework called Formal Concept Analysis (FCA) [8], initially developed by Rudolf Wille. In this framework, data is structured in a tabular format called a formal context, which consists of a set of objects described in terms of their attributes. From this context, we can derive formal concepts, which together form a concept lattice. A concept lattice reveals hierarchical relationships among these concepts and can be used to uncover patterns in the data.

FCA has been applied in various domains, including scientometrics [10], social network analysis [6], and linguistics [15]. In the context of lexical typology [18][14][12][11][9], words are used as objects, the set of meanings also sometimes called frames, as attributes and the relations between the words and meanings form a formal context; its concept lattice, can be interpreted as a semantic map. In the coming sections, we will discuss this approach to semantic mapping in depth, starting with the foundational definitions of formal concept analysis and concept lattices.

1.2.1 Formal Concept Analysis fundamentals

A *formal context* is a triple K = (*G, M, I*), where *G* is a set of objects, *M* is a set of attributes, and *I* is a binary relation indicating which object has which attribute. We define *derivation operators* $(\cdot)'$ for $A \subseteq G$ and $B \subseteq M$ as follows:

$$A' = \{g \in G \mid \forall m \in B\text{:}\ (g, m) \in I\}, \qquad B' = \{m \in M \mid \forall g \in A\text{:}\ (g, m) \in I\}$$

Thus, A' is the set of attributes common to all objects of A and B' is the set of objects sharing all attributes of B.

A *formal concept* of a formal context (G, M, I) is a pair (A, B), where $A \subseteq G$ is the *extent*, $B \subseteq M$ is the *intent*, and $A = B'$ and $B = A'$.

For two formal concepts (A_1, B_1) and (A_2, B_2) of (G, M, I), we say that (A_1, B_1) is a subconcept of (A_2, B_2) and, equivalently, that (A_2, B_2) is a superconcept of (A_1, B_1) iff $A_1 \subseteq A_2$. We use the $\leq$ sign to formalize this relation:

$$(A_1, B_1) \leq (A_2, B_2) :\Leftrightarrow A_1 \subseteq A_2 \ (\Leftrightarrow B_2 \subseteq B_1).$$

The set of all formal concepts derived from a formal context (G, M, I), together with this subconcept–superconcept order relation, form a concept lattice and is denoted by $\mathfrak{B}(G, M, I)$. These lattices are visualized through *line diagrams* (sometimes referred to as *Hasse diagrams*), which shows the hierarchical structure without redundant connections. This lattice structure consists of a natural ordering: concept (A_1, B_1) is at most as general as concept (A_2, B_2) if $A_1 \subseteq A_2$ (equivalently, $B_2 \subseteq B_1$).

The dependencies between attributes and objects can be described by implications. There are two types of implications**:** *attribute* and *object implications*. An attribute implication takes the form: $X \rightarrow Y$, where $X, Y \subseteq M$. We say that this implication *holds* in the context (G,M,I) if $X' \subseteq Y'$, i.e. each object that has all attributes in X also has all attributes in Y. Similarly, an object implication takes the form: $A \rightarrow B$, where $A, B \subseteq G$. We say that this implication *holds* in the context if $A' \subseteq B'$, i.e., every attribute possessed by all objects in A is also possessed by all objects in B.

An implication $X \rightarrow Y$ *follows from* or *is entailed by* a set of implications Σ, if every attribute set $C \subseteq M$ that respects Σ also respects $X \rightarrow Y$. Given a formal context (G,M,I), we say that a set of implications Σ is sound with respect to (G,M,I) if all implications in Σ hold in (G,M,I). It is called complete with respect to (G,M,I) if all implications that hold in (G,M,I), follow from Σ, and it is called non-redundant if no implication in Σ follows from the others. We call an implication set an implication basis of the formal context if it is sound, complete, and non-redundant. The *Duquenne-Guigues (DG)* or *canonical basis* of a context, as defined in [14], has the fewest implications among all implication bases of this context.

1.2.2 Concept Lattice Diagrams

Concept lattices are visualized using *line diagrams* (sometimes referred to as *Hasse diagrams*), which show the hierarchical structure without redundant connections. For two formal concepts (A_1, B_1) and (A_2, B_2) of (G, M, I), we say that (A_1, B_1) is a *proper subconcept* of (A_2, B_2), denoted as $(A_1, B_1) < (A_2, B_2)$, if $(A_1, B_1) \leq (A_2, B_2)$ and $(A_1, B_1) \neq (A_2, B_2)$. We say ($A_1$, B_1) is a *lower neighbour* of (A_2, B_2), denoted as $(A_1, B_1) \prec (A_2, B_2)$, if ($A_1$, B_1) < (A_2, B_2) and there is no formal concept (A, B) of (G, M, I) such that (A_1, B_1) < (A, B) < (A_2, B_2).

In a line diagram of a concept lattice, each formal concept is represented by a node. The nodes are placed so that the node for a concept is always positioned higher than all the nodes for its proper subconcepts. Each concept node is connected by a line to every its lower neighbour.

Nodes are labeled with object and attribute names. We attach each attribute *m* to the node representing its *attribute concept* ($\{m\}'$, $\{m\}''$). Similarly, we attach each object *g* to the node representing its *object concept* ($\{g\}''$, $\{g\}'$).

The edges in the concept lattice diagram express the concept order (the subconcept–superconcept order relation). With the help of the edges, we can read from the diagram which concepts are subconcepts of other concepts and which objects have which attributes. Following *ascending paths* from an object concept, we can get all the attributes of this object, and following the *descending paths* from an attribute concept allows us to find all objects that have this attribute. We can also read the intent and extent of every formal concept in the diagram using this method. Its extent consists of all objects that can be reached from that node by following descending paths, and the intent consists of all attributes reachable by following ascending paths.

1.2.3 Concept Lattices as Semantic Maps

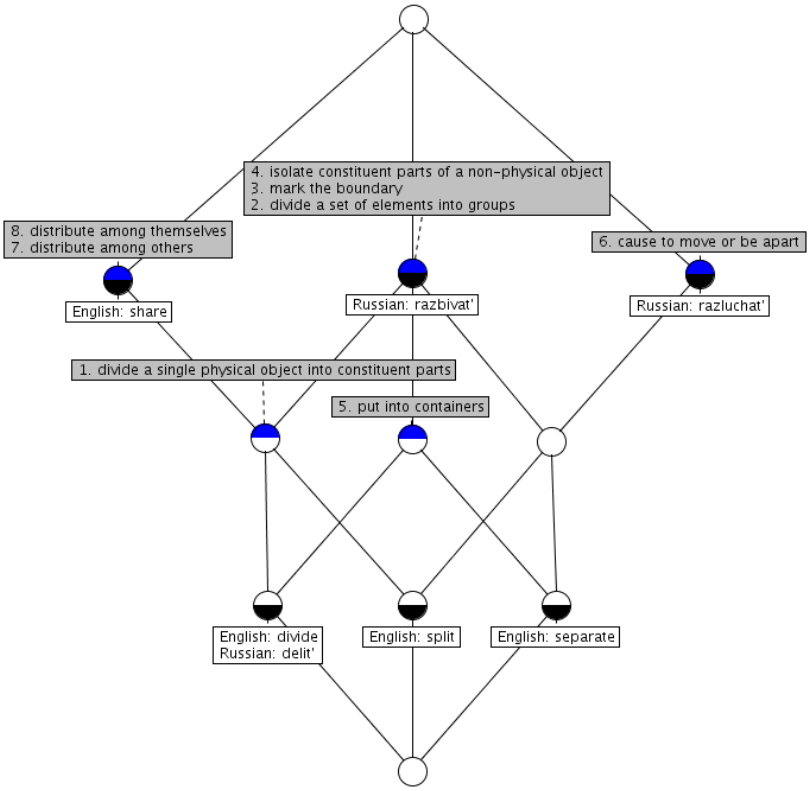
4. isolate constituent parts of a non-physical object
3. mark the boundary
2. divide a set of elements into groups
8. distribute among themselves
7. distribute among others
6. cause to move or be apart
English: share
Russian: razbivat'
Russian: razluchat'
1. divide a single physical object into constituent parts
5. put into containers
English: divide
Russian: delit'
English: split
English: separate

Figure 3. *Concept lattice for the field 'divide' built from data in table 1 (concepts- 11)*

Figure 3 presents a concept lattice constructed from the data in Table 1, where the lexical items function as objects and the semantic meanings serve as attributes. The binary relation between objects and attributes is indicated by 'X' markings. The concept lattice consists of eleven formal concepts, these being represented by the eleven circles in the diagram. Each of the eight attributes is attached to one of the formal concepts and is written slightly above or below the respective circle in a grey

box. The eight objects are represented in the white boxes, each showing a word in English or Russian. Each object and attribute is associated with exactly one formal concept. In some cases where multiple objects or attributes correspond to the same formal concept, they are stacked together and shown within the same box. The concept lattice reveals the hierarchical organization of lexicalization patterns within this semantic domain. Each node represents a specific combination of semantic frames (accessible through upward paths) and the lexical items (accessible through downward paths) that instantiate those frames.

Many other patterns can also be observed from the lattice, for example, the meanings 2, 3, and 4 are attached to the same node, which suggests that they are expressed exactly by the same words in our context. We can also observe implications using the lattice, the attribute concept of the meaning 1 is a proper subconcept of 2,3,4, i.e., $\{1\} \rightarrow \{2, 3, 4\}$, which suggests that every word with the meaning 1 also expresses 2, 3, and 4. On the other hand, meanings 2,3,4 can be combined with 6 without having to express 1 (as in the word "separate"); so, $\{2, 3, 4, 6\} \rightarrow \{1\}$ doesn't hold.

These implications derived from the concept lattice hold within our specific formal context, which contains only a small sample of English and Russian lexemes. However, critical questions remain: Do these implications generalize to the complete lexicons of these languages? Are there counterexamples in other languages? These are precisely the questions linguists must address when constructing semantic maps. Attribute exploration provides a systematic methodology to ensure no such questions are overlooked. This interactive procedure generates explicit questions for each implication, for example, asking whether there exists a word that expresses meaning A without expressing meaning B and uses the answers to either confirm the implication or identify counterexamples that necessitate revision of the semantic structure. Yet a practical challenge remains: how can researchers efficiently answer these questions across numerous languages without extensive manual investigation? This motivates our exploration of using large language models as an efficient tool for validating implications and identifying potential counterexamples across diverse languages.

In the following sections, we will introduce attribute exploration, which is part of the Conceptual Exploration framework, and then present the interface we developed that uses LLMs to facilitate these explorations.

2 Collecting data for semantic maps using Conceptual Exploration

Before discussing concept exploration [7], we will first examine the traditional approaches used to collect data for constructing classical semantic maps.

2.1 Traditional linguistic data collection methods

Traditionally linguists follow two approaches to build semantic maps, the onomasiological approach and the semasiological approach; both approaches can be used independently or combined [5, 10, 20]. In the onomasiological *(or top-down)* approach, linguists begin by selecting a specific semantic or functional domain and identifying its core meanings. They then look at how these meanings are expressed in different languages using particular forms like words or constructions. The semasiological *(or bottom-up)* approach begins with language-specific words or constructions and examines all the meanings they can have within that language.

According to [10], most studies begin with an onomasiological approach by selecting a particular domain and identifying the core meanings in this domain, and search for the individual forms that expresse these meanings across various languages. This is typically followed by a semasiological

analysis, where a lexical matrix is used to document all the meanings associated with each linguistic form. Multiple entries in a single row indicate polysemy, while multiple entries in a column suggest synonymy or translational equivalence among the forms.

Both approaches can be implemented through structured questionnaires with native speakers. For the onomasiological approach, researchers present speakers with specific concepts or scenarios and ask them to provide the appropriate words in their language. For the semasiological approach, researchers give speakers particular words and ask them to explain all the different meanings or contexts where these words can be used. These questionnaire methods provide direct access to speaker intuitions about word meanings and help identify colexification patterns that form the basis of semantic maps.

2.2 Conceptual Exploration

Conceptual exploration [7] is a knowledge-driven procedure in Formal Concept Analysis that supports the systematic completion of a formal context through interaction with a domain expert. This approach guides the expert through a series of questions aimed at confirming or rejecting potential implications between attributes or objects. Conceptual exploration combines two main approaches: attribute exploration and object exploration, addressing different aspects of knowledge completion within the conceptual exploration framework.

In attribute exploration, we start with context (E, M, J), which can be considered as a subcontext of a larger domain (G, M, I). For this subcontext, we generate the implication basis; next we take one implication from this basis and present it to a domain expert in the form of a question: "If an object has attribute X, does it necessarily have attribute Y as well?"; the expert either confirms this implication or refutes it by providing a counterexample object that is then added to the context, which may change the previous implication basis. The process terminates when all the implications are either entailed by the context or are confirmed.

For object exploration, we can transpose the initial context (E, M, J) to (M, E, J^{-1}) and perform the attribute exploration on this transposed context. In this setting, the exploration algorithm poses questions of the form: *"Within your domain, is it always true that any attribute that applies to all objects in set A also applies to all objects in set B?"* Here, A and B are subsets of the object set G in the original context (now treated as attributes in the transposed context). If the expert confirms such a statement, it is accepted as a valid implication in the object domain; otherwise, a counterexample is provided—an attribute that applies to all objects in A but not to all objects in B.

Attribute exploration and object exploration can be used to support, respectively, the top-down and bottom-up approaches discussed in the previous section. When we take lexical data such as one shown in Table 1 as our initial context, attribute exploration generates attribute implications $X \rightarrow Y$, where X and Y are frames (meanings). For each implication, we pose questions to the domain expert in the form: "Does every word that conveys meaning X also conveys meaning Y?" The expert either confirms the validity of this implication or provides counterexamples—words that convey X but not Y.

Similarly, object exploration supports the semasiological approach by generating object implications $A \rightarrow B$, where A and B are words (linguistic forms). For each implication, we pose questions to the domain expert in the form: "Does every meaning conveyed by all words in set A also get conveyed by all words in set B?" The expert either confirms the validity of this implication or provides counterexamples meanings that apply to all words in A but not to all words in B.

3 LLM Supported Conceptual Exploration

Large Language Models (LLMs) can potentially assist in conceptual exploration by acting as domain experts while exploring linguistic data. LLMs have access to cross-linguistic information that may be useful for validation of implications. Also, because LLMs take information from many sources, they may use words as they are actually used in natural conversations and may allow us to deal with the subjectivity issue related to expert validation.

This section presents an LLM-assisted system designed to facilitate interactive conceptual exploration through automated content processing and flexible user interaction modes. The system allows researchers to upload documents and engage with the content through multiple exploration pathways, from manual analysis to fully automated processing. By leveraging LLM capabilities, the system aims to understand context, identify conceptual relationships, and respond to user queries in natural language, making conceptual exploration more intuitive and comprehensive.

3.1 System Architecture

The LLM-assisted concept exploration system employs a modular architecture that facilitates both attribute and object exploration through three distinct interaction modes: **Manual Mode** for direct user control, **Assisted Mode** for guided exploration with LLM support, and **Auto Mode** for fully autonomous analysis.

This study builds upon code from the **FCATNG** library (https://github.com/tomhanika/fcatng). We utilized the library's existing implementation of the implication generation algorithm without modification, while independently developing other components of the system.

The system accepts context input in multiple formats (see Appendix B). Once the input file is successfully uploaded, a Python object is created. The user can then choose between attribute or object exploration pathways, each supporting all three interaction modes.

In **Manual Mode**, users select a specific implication from the set of all implications generated from the context for evaluation. Users can either confirm or reject the selected implication. If confirmed, the implication is added to the confirmed implications list and removed from the set of all implications. If rejected, a counterexample must be provided. This counterexample is validated in the background by the system based on certain rules (refer section 3.3), and if the counterexample is valid, it is then added to the context.

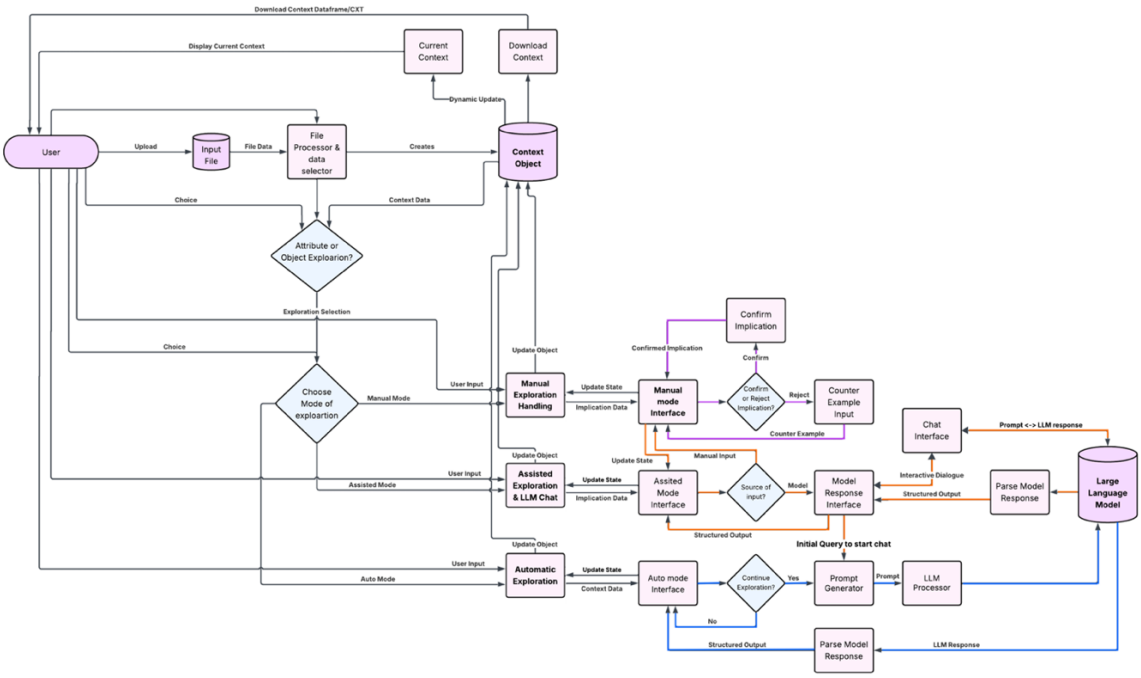

Figure 4. *The dataflow diagram of the system*

In **Assisted Mode**, users begin by selecting a specific implication from the generated set, similar to Manual Mode. Upon selection, users can click the "Ask the Model" button, which triggers an API call using a predefined prompt template to retrieve structured JSON responses. This output provides comprehensive analysis including the model's assessment of whether the implication is valid or invalid. For invalid implications, the response includes a counterexample, a detailed explanation of why this counterexample disproves the implication, and a sentence demonstrating the counterexample's usage.

Following an interaction with the model in the chat interface, users may receive a response (such as a counterexample) they wish to process further. To facilitate this, they can issue a command (e.g., *"Please parse the previous result to the system"*) prompting the system to interpret the model's most recent output. The system then updates the implication evaluation accordingly. Users can choose to accept this parsed result, which either adds the validated implication to the verified set or integrates the counterexample into the context. This process mirrors the validation and integration workflow of Manual Mode while incorporating the advantages of AI-assisted analysis and explanation.

In **Auto Mode**, exploration initiates from the first implication generated from the canonical basis. The system requests two user inputs: a system prompt containing configuration parameters (such as target languages for counterexamples) and the maximum number of implications to process before termination. Exploration terminates upon either reaching this specified limit or exhausting all available implications.

The system employs an internal generator that yields the current implication from the context, which is then integrated with the prompt template and transmitted to the LLM agent. The structured JSON response contains the same output format as Assisted Mode: validity assessment, counterexamples, explanations, and usage sentences.

Upon receiving the LLM response, the system performs automated validation. If the response satisfies all system constraints without conflicts, processing advances to the next implication. When conflicts arise (see section 3.3), the system resubmits the query with explicit conflict notification. This retry mechanism allows up to three attempts per implication. If conflicts persist after the third attempt, the system skips the problematic implication and proceeds to the next implication.

Upon exploration completion, the system generates comprehensive logs documenting all processing steps, including confirmed implications, skipped implications with associated conflict reasons, and the final state of the knowledge base. This audit trail enables users to review the autonomous exploration process and validate the system's decision-making logic.

The three interaction modes function as a unified exploration framework, enabling users to select the appropriate level of automation based on their expertise, dataset size, and research objectives. Manual Mode provides complete control for domain experts requiring precise validation, Assisted Mode combines human insight with AI capabilities for balanced exploration, and Auto Mode enables automated processing of implications. This modular design allows seamless transitions between modes during a single exploration session, supporting iterative refinement strategies where users can begin with automated processing and switch to manual validation for critical implications. The consistent underlying validation and integration mechanisms across all modes ensure data integrity and maintain compatibility with standard FCA theoretical frameworks throughout the exploration process.

3.2 User Interface design

The user interface is implemented using **Shiny for Python** (https://shiny.posit.co/py/), leveraging its reactive programming model to create a dynamic and responsive web application. The interface design prioritizes intuitive navigation and seamless workflow integration across all three interaction modes. Shiny's component-based architecture enables modular UI development with real-time updates and server-side processing integration.

A critical feature of the system is its ability to render interface elements dynamically without page resets or session interruptions. The context data remains persistent throughout the entire user session, maintained through Shiny's reactive state management system. This ensures that all user interactions, from data input through exploration completion, preserve the evolving context and exploration history without loss of information or interface state.

3.2.1 Main Dashboard and Data Input Interface

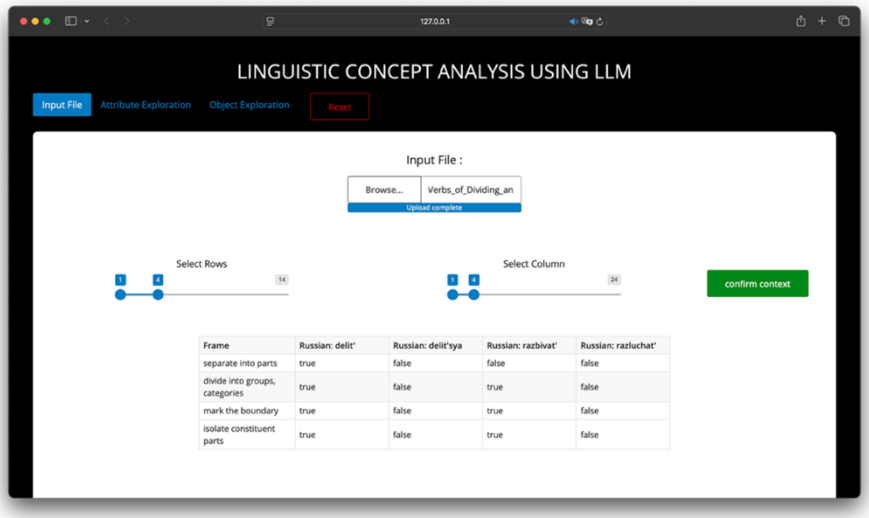

Figure 5. *System Landing Page*

The primary interface presents a clean dashboard for data input and configuration. Users upload context files through Shiny's file input widget supporting CSV, CXT, and Excel formats. The data preprocessing section features slider widgets for column and row selection with live preview using Shiny's data table output for verification before exploration.

3.2.2 Exploration Type and Mode Selection

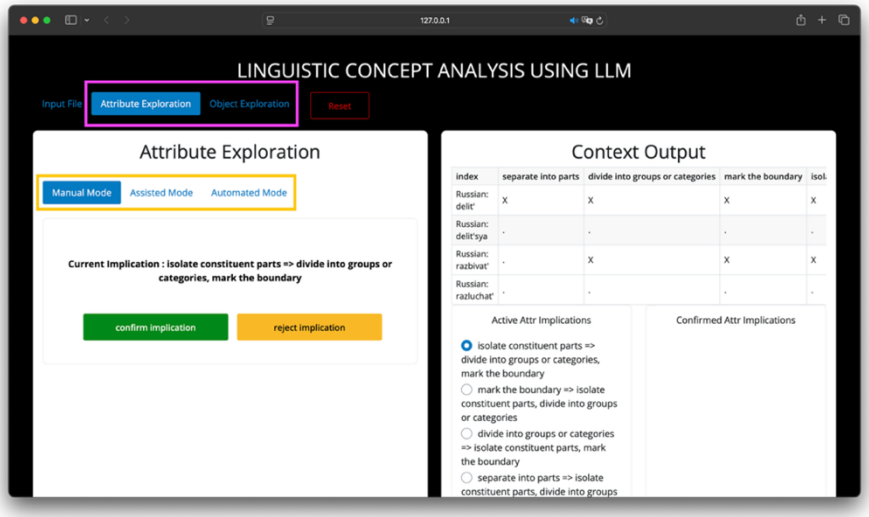
127.0.0.1
LINGUISTIC CONCEPT ANALYSIS USING LLM
Input File
Attribute Exploration
Object Exploration
Reset
Attribute Exploration
Manual Mode
Assisted Mode
Automated Mode
Current Implication : isolate constituent parts => divide into groups or categories, mark the boundary
confirm implication
reject implication
Context Output
index
separate into parts
divide into groups or categories
mark the boundary
Russian: delit'
Russian: delit'sya
Russian: razbivat'
Russian: razluchat'
Active Attr Implications
isolate constituent parts => divide into groups or categories, mark the boundary
mark the boundary => isolate constituent parts, divide into groups or categories
divide into groups or categories => isolate constituent parts, mark the boundary
separate into parts => isolate constituent parts, divide into groups
Confirmed Attr Implications

Figure 6. *Interface for Attribute Exploration*

The top panel contains buttons for selecting between object and attribute exploration types (highlighted in pink). Once a type is selected, a secondary panel appears, allowing users to choose one of three interaction modes (highlighted in yellow): Manual, Assisted, or Auto.

3.2.3 Manual Mode Interface

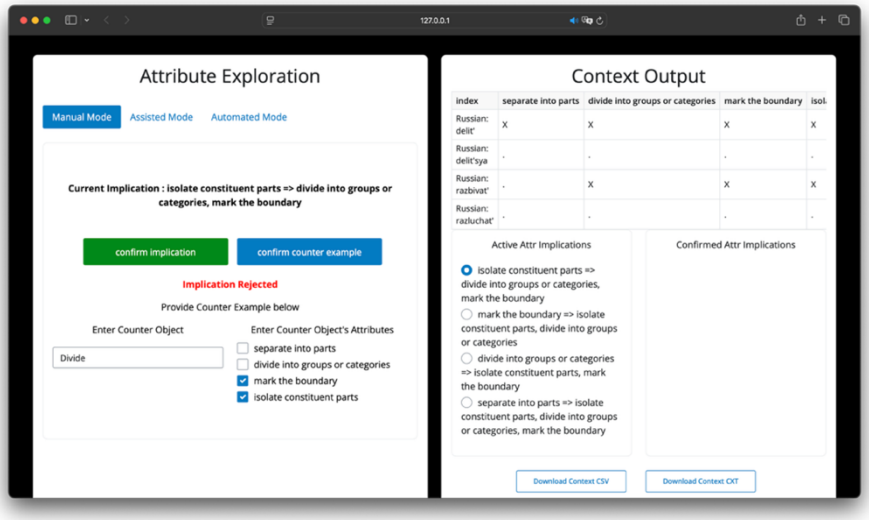
127.0.0.1
Attribute Exploration
Manual Mode
Assisted Mode
Automated Mode
Current Implication : isolate constituent parts => divide into groups or categories, mark the boundary
confirm implication
confirm counter example
Implication Rejected
Provide Counter Example below
Enter Counter Object
Divide
Enter Counter Object's Attributes
separate into parts
divide into groups or categories
mark the boundary
isolate constituent parts
Context Output
index
separate into parts
divide into groups or categories
mark the boundary
Russian: delit'
Russian: delit'sya
Russian: razbivat'
Russian: razluchat'
Active Attr Implications
isolate constituent parts => divide into groups or categories, mark the boundary
mark the boundary => isolate constituent parts, divide into groups or categories
divide into groups or categories => isolate constituent parts, mark the boundary
separate into parts => isolate constituent parts, divide into groups or categories, mark the boundary
Confirmed Attr Implications
Download Context CSV
Download Context CXT

Figure 7. *Manual Mode of the Attribute Exploration*

Features a side panel displaying the complete set of implications for selection. The selected implication appears in the main panel with accept/reject action buttons. Upon rejection, the system prompts for counter-example input through a modal dialog, followed by attribute selection widgets for the counterexample object.

3.2.4 Assisted Mode Interface

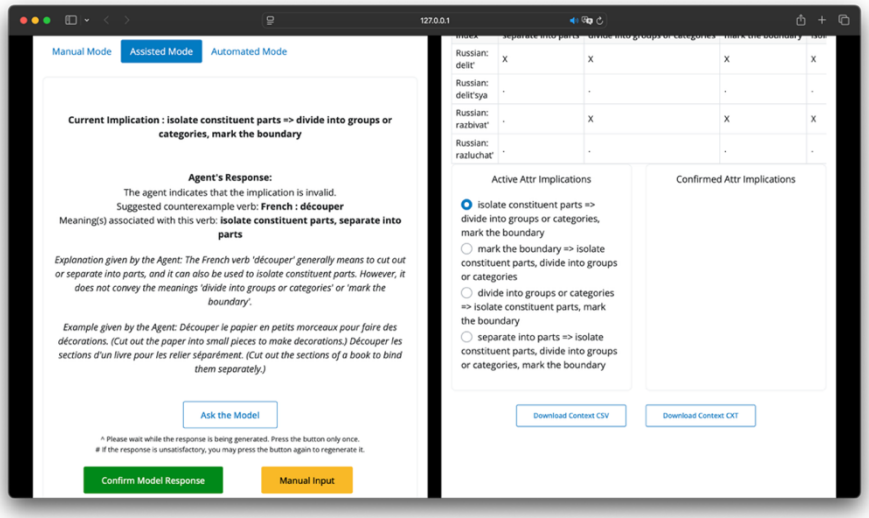
127.0.0.1
Manual Mode
Assisted Mode
Automated Mode
Current Implication : isolate constituent parts => divide into groups or categories, mark the boundary
Agent's Response:
The agent indicates that the implication is invalid.
Suggested counterexample verb: French : découper
Meaning(s) associated with this verb: isolate constituent parts, separate into parts
Explanation given by the Agent: The French verb 'découper' generally means to cut out or separate into parts, and it can also be used to isolate constituent parts. However, it does not convey the meanings 'divide into groups or categories' or 'mark the boundary'.
Example given by the Agent: Découper le papier en petits morceaux pour faire des décorations. (Cut out the paper into small pieces to make decorations.) Découper les sections d'un livre pour les relier séparément. (Cut out the sections of a book to bind them separately.)
Ask the Model
* Please wait while the response is being generated. Press the button only once.
If the response is unsatisfactory, you may press the button again to regenerate it.
Confirm Model Response
Manual Input
Russian: delit'
Russian: delit'sya
Russian: razbivat'
Russian: razluchat'
Active Attr Implications
isolate constituent parts => divide into groups or categories, mark the boundary
mark the boundary => isolate constituent parts, divide into groups or categories
divide into groups or categories => isolate constituent parts, mark the boundary
separate into parts => isolate constituent parts, divide into groups or categories, mark the boundary
Confirmed Attr Implications
Download Context CSV
Download Context CXT

Figure 8. *Assisted Mode of the Attribute Exploration*

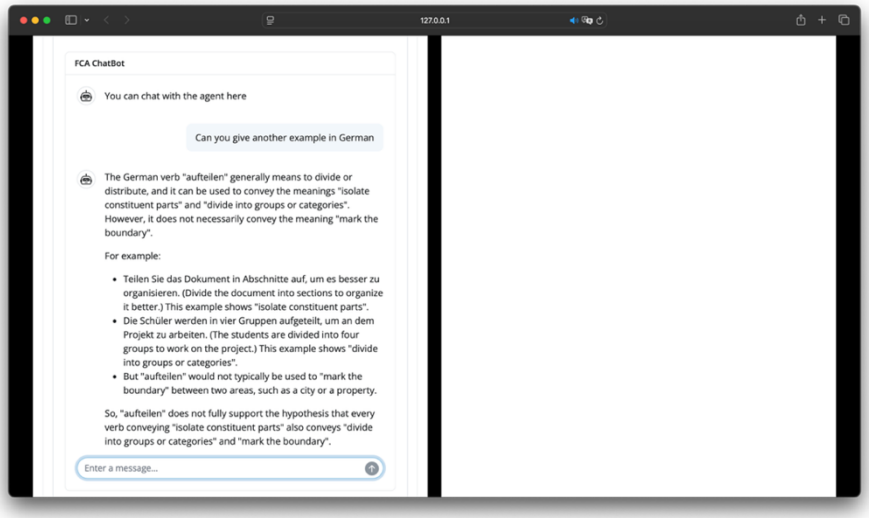
127.0.0.1
FCA ChatBot
You can chat with the agent here
Can you give another example in German
The German verb "aufteilen" generally means to divide or distribute, and it can be used to convey the meanings "isolate constituent parts" and "divide into groups or categories". However, it does not necessarily convey the meaning "mark the boundary".
For example:
Teilen Sie das Dokument in Abschnitte auf, um es besser zu organisieren. (Divide the document into sections to organize it better.) This example shows "isolate constituent parts".
Die Schüler werden in vier Gruppen aufgeteilt, um an dem Projekt zu arbeiten. (The students are divided into four groups to work on the project.) This example shows "divide into groups or categories".
But "aufteilen" would not typically be used to "mark the boundary" between two areas, such as a city or a property.
So, "aufteilen" does not fully support the hypothesis that every verb conveying "isolate constituent parts" also conveys "divide into groups or categories" and "mark the boundary".
Enter a message...

Figure 9. *Chat interface in the assisted mode of the Attribute Exploration*

Extends Manual Mode with AI interaction. Implications are selected from the side panel set. The "Ask the Model" button triggers API calls for AI recommendations on the selected implication. Users can then accept the AI response, provide manual input, or engage in chat dialogue with the model through the integrated chat interface.

3.2.5 Auto Mode Interface

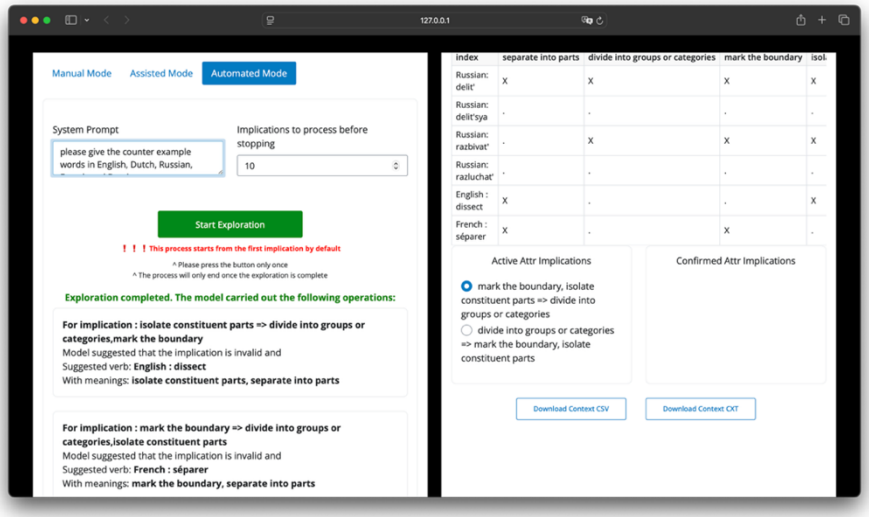
127.0.0.1
Manual Mode
Assisted Mode
Automated Mode
System Prompt
please give the counter example words in English, Dutch, Russian,
Implications to process before stopping
10
Start Exploration
! ! ! This process starts from the first implication by default
^ Please press the button only once
^ The process will only end once the exploration is complete
Exploration completed. The model carried out the following operations:
For implication : isolate constituent parts => divide into groups or categories,mark the boundary
Model suggested that the implication is invalid and
Suggested verb: English : dissect
With meanings: isolate constituent parts, separate into parts
For implication : mark the boundary => divide into groups or categories,isolate constituent parts
Model suggested that the implication is invalid and
Suggested verb: French : séparer
With meanings: mark the boundary, separate into parts
index
separate into parts
divide into groups or categories
mark the boundary
Russian: delit'
Russian: delit'sya
Russian: razbivat'
Russian: razluchat'
English : dissect
French : séparer
Active Attr Implications
mark the boundary, isolate constituent parts => divide into groups or categories
divide into groups or categories => mark the boundary, isolate constituent parts
Confirmed Attr Implications
Download Context CSV
Download Context CXT

Figure 10. *Auto mode of the Attribute Exploration*

The system requires two user inputs: text area inputs for system prompts and numeric inputs for processing limits. Upon completion, the system displays comprehensive logs detailing all exploration steps taken during the automated process.

3.2.6 Download Functionality

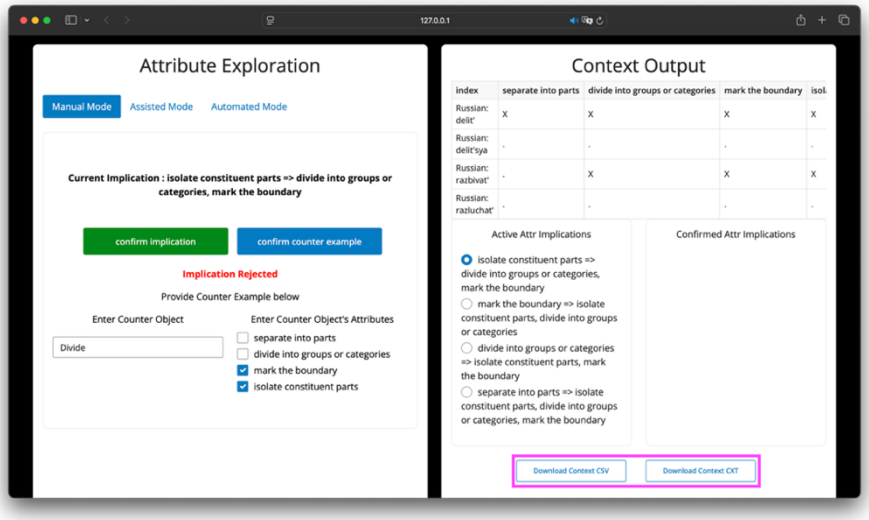
127.0.0.1
Attribute Exploration
Manual Mode
Assisted Mode
Automated Mode
Current Implication : isolate constituent parts => divide into groups or categories, mark the boundary
confirm implication
confirm counter example
Implication Rejected
Provide Counter Example below
Enter Counter Object
Enter Counter Object's Attributes
Divide
separate into parts
divide into groups or categories
mark the boundary
isolate constituent parts
Context Output
index
separate into parts
divide into groups or categories
mark the boundary
Russian: delit'
Russian: delit'sya
Russian: razbivat'
Russian: razluchat'
Active Attr Implications
isolate constituent parts => divide into groups or categories, mark the boundary
mark the boundary => isolate constituent parts, divide into groups or categories
divide into groups or categories => isolate constituent parts, mark the boundary
separate into parts => isolate constituent parts, divide into groups or categories, mark the boundary
Confirmed Attr Implications
Download Context CSV
Download Context CXT

Figure 11. *Download functionality of the interface*

Features two download buttons for exporting the current context: CSV format and CXT file format (a standard format for representing formal contexts in FCA), both implemented using Shiny's download button widgets with server-side file generation.

3.3 Error Handling Mechanisms

To ensure the reliability of response generation, the system implements a comprehensive error handling framework with three primary validation mechanisms that work together to maintain logical consistency and structural compliance. First, the system enforces **uniqueness of counterexamples**. If the model suggests a counterexample that has already been processed or exists in the list of previously analyzed examples, the system detects this repetition and engages the model in a dialogue loop. A follow-up prompt is sent requesting a different counterexample that has not been previously considered, thereby preserving the novelty and usefulness of each test case.

Second, the system verifies whether the counterexample logically invalidates the hypothesis as defined in Formal Concept Analysis. For a hypothesis of the form $X \rightarrow Y$, a valid counterexample must involve an object (i.e., a word) that possesses **all attributes in set X (the premise)** but **lacks at least one attribute from set Y (the conclusion)**. If the model returns an object that either fails to include all premise attributes or includes all conclusion attributes, the system identifies this logical inconsistency and initiates a corrective prompt. This mechanism ensures that only valid counterexamples are generated from the model, which can then be included in the context.

Third, all model responses are expected in a strict JSON format containing predefined keys such as "output", "verb", "meaning", "language", "explanation", and "example". If the model fails to include any required fields or returns an improperly formatted JSON object, the system prompts the

model to revise the output. These checks are enforced both in assisted and auto mode. These mechanisms collectively ensure that only valid, unique, and properly formatted counterexamples are accepted for analysis.

The system applies these three validation strategies with a maximum of three attempts. On each attempt, the model is prompted to correct any detected errors, whether they concern counterexample uniqueness, logical validity, or JSON-format compliance. If, after three correction cycles, the model still fails to produce a valid response, the system terminates the process for the current implication and skips it, thus avoiding infinite correction loops.

4 Experimental Setup and Results

This section presents the experimental setup used to assess the effect of various prompting strategies on LLM in the context of attribute and object exploration and later presents the results of these experiments.

4.1 Experimental Setup

Datasets. We used the dataset from the semantic field "divide," as introduced in Section 1, with two configurations:

For attribute exploration:

- **4×4 Context**: 4 verbs and 4 frames (last four rows that contain English words and first four columns from Table 1)
- **4×8 Context**: 4 verbs and 8 frames (last four rows that contain English words and all columns from Table 1)

For object exploration:

- **4×4 Context**: same as for attribute exploration.
- **7×4 Context**: 7 verbs and 4 frames (all rows and first four columns from Table 1)

Large Language Model. As this is a pilot study, we conducted all experiments using a single model: meta-llama/Llama-3.3-70B-Instruct.

Prompting Strategy. To assess how prompt design affects LLM performance in validating semantic implications, we defined three prompting strategies, each incrementally incorporating more structure and guidance. These strategies were applied for the same initial context and for all the above models, all prompts request **structured JSON output**.

- **S1 – Minimal Prompt -** represents the simplest formulation of the task. It provides only the hypothesis to be tested and specifies the required JSON output format. This version contains a meanings list (or words list for object exploration) but does not have any background context, structured steps, or reasoning instructions. Its purpose is to establish a baseline by testing how the system performs when given only the essential information needed to generate a response.

- **S2 – Structured Prompt -** expands on the minimal version by introducing additional context and a more explicit procedure. In addition to the meanings list, a clearer hypothesis

statement, and stepwise instructions on how to evaluate verbs against the hypothesis. The output specification is more detailed than in S1, but it remains relatively lightweight, focusing on factual reporting rather than interpretive reasoning.

- **S3 – Context-Augmented Prompt –** This is the most elaborate version. It provides extensive contextual information, such as examples and already checked verbs, and includes constraints on how the system should process and report results. The JSON output requires not only a decision but also an explanation, language identification, and usage examples. This design encourages the model to demonstrate deeper reasoning and justification, making it the richest prompt type for testing interpretive and analytical capabilities.

The exact prompts for each strategy are provided in the appendix C.

Evaluation Parameters. we evaluate the system based on following parameters:

- **Implications Checked:** Total number of implications tested during exploration
- **Implications Confirmed:** Implications accepted as valid without counterexamples
- **Implications Skipped:** Implications abandoned after three consecutive failed attempts to generate valid counterexamples
- **Counter Examples Provided:** Total number of unique valid counterexamples generated during exploration
- **Invalid Counter Example Error*:** Counterexample fails to satisfy the premise requirements of the implication being tested
- **Repeated Verb Error*:** System provides a counterexample already present in the formal context

Experimental Procedure. All experiments were conducted using the automated mode of the developed system. For each configuration, we systematically tested all three prompting strategies (S1, S2, and S3) across both dataset contexts. To account for the randomness inherent in large language models, each experimental configuration was repeated ten times. This includes ten independent runs for:

- Each prompting strategy (S1, S2, S3)
- Each initial context type (4×4 and 4×8 for attributes; 4×4 and 8×4 for objects)
- Both exploration modes (attribute and object exploration)

Additionally, we performed a sequential exploration experiment, where attribute exploration was followed by object exploration on the same context. For this sequential mode, we used only the S2 prompting strategy, as preliminary observations indicated it performed slightly better than S1 and S3 in balancing structure and flexibility.

All reported results are presented as mean ± standard deviation, with values rounded to the nearest integer, computed over the ten independent runs for each configuration. This approach ensures that our findings account for the stochastic nature of LLM outputs while providing robust statistical measures of system performance.

4.2 Evaluation and Results

This section presents and analyzes the results of our experiments. Quantitative performance metrics from all independent runs are reported in tables, along with some selected concept lattices generated from these explorations.

4.2.1 Attribute Exploration

	No of Implications Checked	Implications Skipped	Implications Confirmed	Counter Examples Provided	Invalid Counter Example Error	Repetition Error
S1	8 ± 1	0 ± 0	0 ± 0	8 ± 1	0 ± 0	3 ± 2
S2	7 ± 1	0 ± 0	1 ± 0	6 ± 1	0 ± 0	1 ± 2
S3	8 ± 1	0 ± 0	0 ± 1	7 ± 0	0 ± 0	0 ± 0

Table 2a. *Results of Attribute Exploration using the 4×4 context as the initial content*

	No of Implications Checked	Implications Skipped	Implications Confirmed	Counter Examples Provided	Invalid Counter Example Error	Repetition Error
S1	55 ± 7	14 ± 3	7 ± 1	34 ± 6	20 ± 4	57 ± 12
S2	29 ± 10	2 ± 2	7 ± 2	20 ± 7	2 ± 1	25 ± 14
S3	67 ± 7	1 ± 1	10 ± 2	56 ± 7	3 ± 1	17 ± 5

Table 2b. *Results of Attribute Exploration using the 4×8 context as the initial content*

For the first experiment using the 4×4 context as the initial content, S1 produced a 100% implication rejection rate across all 83 implication tests, with zero confirmed implications, S2 and S3 also were able to confirm 1 or 2 implications occasionally. All the three prompts expanded vocabulary to include English counterexamples, but occasional generated non-English counterexamples, for example S2 produced, French: *délimiter*, *découper*, and German: *zerteilen*. Each run identified 10–14 distinct lexical items, collectively mapping a wide semantic domain that included general separation verbs (*separate*, *split*, *divide*), technical or formal terms (*dissect*, *delineate*, *demarcate*, *compartmentalize*), physical action verbs (*chop*, *cut*, *slice*, *cleave*), and abstract cognitive verb (*analyze*, *categorize*, *classify*, *partition*) .

In the second experiment, using the 4×8 context as the initial content, the system consistently generated words expanded the depth and nuance within the same categories like before, but they generated more specific physical verbs (mince, shred, fracture, saw), additional distribution nuances (pack, fill, disperse), slightly stronger boundary verbs (sever, disjoint), and included multilingual terms mapping the same concepts across languages. The larger context likely allowed longer exploration, enabling the LLM to discover these additional words.

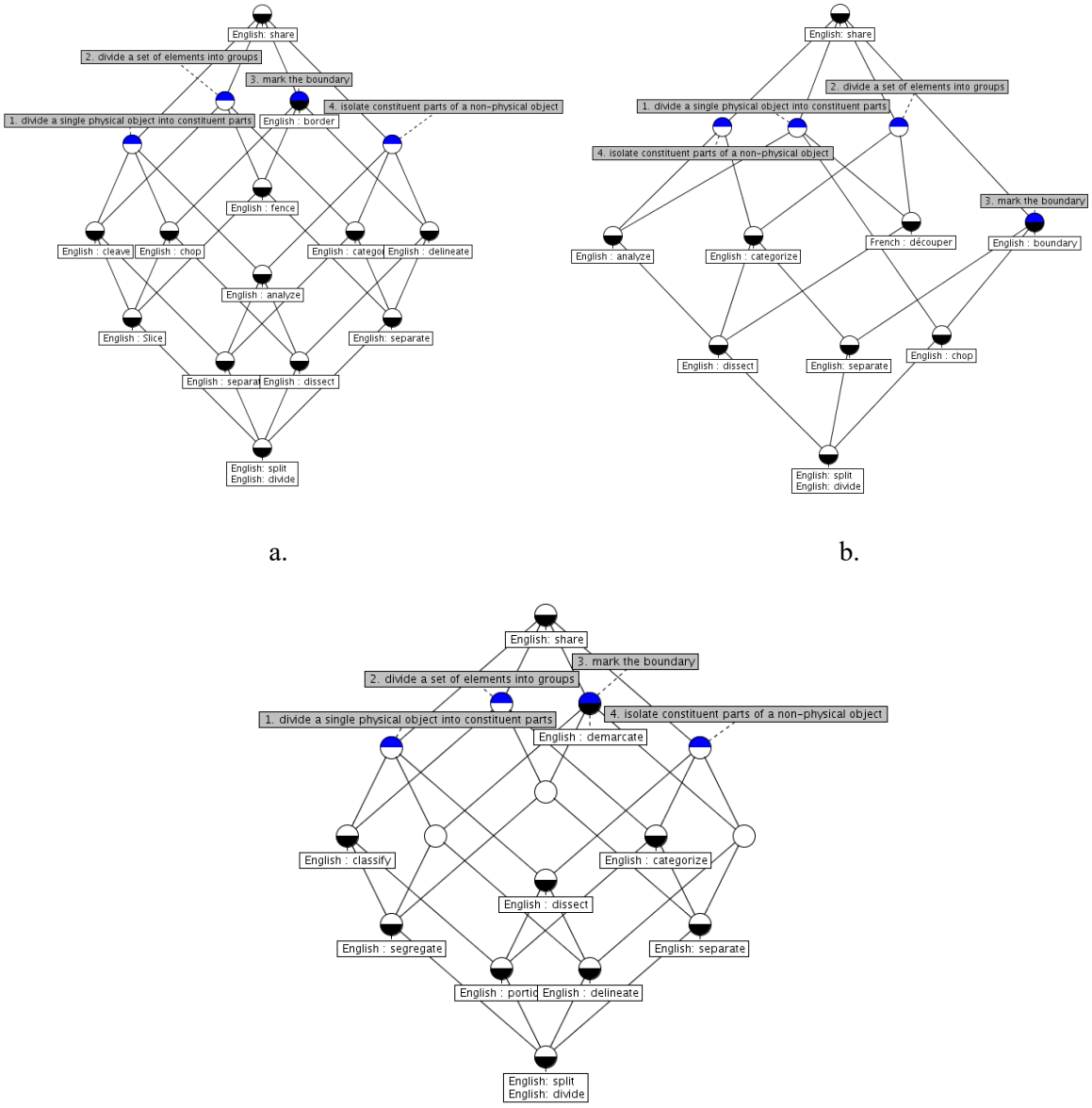

Figure 12. *Concept lattices obtained from one of the ten Attribute Exploration runs using the 4×4 context as the initial content for each prompting strategy: (a) S1 (concepts: 16), (b) S2 (concepts: 12), (c) S3 (concepts: 16)*

Figure 12 presents the concept lattices generated following explorations staring from *the* 4×4 context using different prompts. For these particular runs, S1 and S3 did not confirm any implications, while S2 confirmed "*mark the boundary*, *divide a set of elements into groups → isolate constituent parts of a non-physical objec*t" and "*mark the boundary*, *isolate constituent parts of a non-physical object → divide a set of elements into groups*".

Examining the word "English: dissect" reveals that it consistently associates with meanings 1 and 4 across all three lattices; however, its other semantic associations vary, with S1 additionally including

meaning 3 and S2 including 2. This inconsistency likely occurred because the same word can be given as counterexample to different implications during the different exploration runs.

Notably, some patterns remain consistent across explorations. The object concept of "English: categorize" consistently appears as a superconcept of the object concept of "English: separate" in all three lattices. While the semantic frames (meanings) linked to these words differ across lattices, "English: categorize" is consistently regarded as a more specific term compared to "English: separate", since the set of meanings of the former is a subset of that of the latter.

Despite being more concise than S3, S2 demonstrated better performance, as it confirmed more implications proportionally and had similar error counts as S3 (refer Table 2a and 2b). We hypothesize that for attiribute exploration, S3's extensive instructions may overload the model with constraints, leading to overly conservative responses or shallow pattern matching rather than genuine reasoning. S2 appears to strike a better balance, providing sufficient structure to guide the task while maintaining enough flexibility for effective semantic reasoning. This suggests that optimal prompt design involves finding a middle ground between minimal guidance (S1) and excessive elaboration (S3). Overall, while structured prompts clearly reduce error counts, the question of whether they enhance the system's underlying logical reasoning abilities, as opposed to simply constraining error-prone outputs, remains open.

4.2.2 Object Exploration

	No of Implications Checked	Implications Skipped	Implications Confirmed	Counter Examples Provided	Invalid Counter Example Error	Repetition Error
S1	2 ± 0	0 ± 0	1 ± 0	1 ± 0	1 ± 0	0 ± 0
S2	2 ± 0	0 ± 0	1 ± 0	1 ± 0	1 ± 0	0 ± 0
S3	2 ± 0	0 ± 0	1 ± 0	1 ± 0	1 ± 1	0 ± 0

Table 3a. Results of Object Exploration using the 4×4 context as the initial content

	No of Implications Checked	Implications Skipped	Implications Confirmed	Counter Examples Provided	Invalid Counter Example Error	Repetition Error
S1	22 ± 4	4 ± 2	4 ± 1	14 ± 3	18 ± 4	11 ± 4
S2	20 ± 3	2 ± 1	6 ± 1	12 ± 4	12 ± 3	10 ± 3
S3	24 ± 4	4 ± 2	6 ± 2	13 ± 3	5 ± 3	17 ± 8

Table 3b. Results of Object Exploration using the 7×4 context as the initial content

S1	English:divide, English:split, English:share, English:separate, Russian:razluchat', Russian:delit' → Russian:delit'sya English:divide, English:split, Russian:razbivat', English:share, Russian:razluchat', Russian:delit', Russian:delit'sya → English:separate
S2	English:split, Russian:delit', Russian:razbivat', English:divide, English:separate, Russian:razluchat' → Russian: delit'sya English:divide, English:share, English: split, Russian: delit' → Russian: razbivat' Russian:delit'sya, English:split, Russian:delit', English:divide, Russian:razluchat' → Russian: razbivat'
S3	English:share, English:divide, English:separate, Russian:razbivat', English:split, Russian:delit' → Russian: delit'sya

Table 4. Confirmed implications from *one of the ten Object Exploration runs using the 8×4 context as the initial content for each prompting strategy*

For the 4×4 context, almost all the exploration for all prompt types ended after evaluation a single implication, "*divide, share, split => separate*", the implication was rejected almost every time. The meanings discovered were largely consistent across different runs and were related to "***distribute among others***". Some variants are "to give part of something to someone", "to distribute or dispense portions of something" and "to give part of something to someone else".

For the 7×4 context, S1 and S2 broadly discovered meanings corresponding to two frames from Table 1: "***distribute among others***" and "***divide a single physical object into constituent parts***". Meanings such as "*to give part of something to someone*", "*to give part of something to somebody*", "*to give part of something to someone else*" and "*to share or portion out something*" were discovered which can be semantically related to "*distribute among others*", and meaning such as "*to separate or part things*", "*to separate or part something into two or more parts*" and "*to separate or part things that are connected*" can be semantically related to "*divide a single physical object into constituent parts*". From Table 4, we observe that the implication *"English:divide, English:split, English:share, English:separate, Russian:razluchat', Russian:delit' → Russian:delit'sya"* was accepted by all three prompting strategies (S1, S2, and S3).

S3 in addition to the previous frames, was also able to discover meanings like "*to move or cause something to move from one place to another*", "*to move or force something apart*" which can be semantically related to "*cause to move or be apart*". It also discovered meanings such as "*to set or determine the boundaries or limits of something*" which can be related to "*mark the boundary*" and "*to categorize or classify something*" which can be related to "*divide a set of elements into groups*".

Redundancy in results likely arose from the system's inability to recognize semantic equivalence across paraphrases. Without mechanisms to measure semantic similarity, the generation process produced multiple linguistic variants of essentially identical meanings, treating them as distinct discoveries. For example, phrases like "to give part of something to someone," "to somebody," and "to someone else" were all considered separate meanings, as were "to give part of something as a proportion of a whole" and "to give part of something to somebody." This reveals a fundamental limitation: while the system can generate semantically related expressions, it lacks the meta-semantic

capability to consolidate equivalent meanings or systematically map the semantic space without redundant revisiting.

None of the three prompts were useful to discover meanings which could be related with "*distribute among themselves*" or "*put into containers*". This systematic omission may be due the presence of a potential bias in the model's underlying training distribution, the LLM tends to favor highly frequent or canonical semantic associations of *divide*, rather than exhaustively probing less common or more specialized semantic dimensions. Consequently, the model's semantic coverage appears constrained by the prominence of patterns present in its training corpus, which may limit its effectiveness in uncovering rarer conceptual variants.

4.2.3 Sequential Attribute and Object Exploration

For the final experiment, we performed sequential exploration using the S2 prompting strategy, beginning with the 4×4 formal context. The results were similar to the earlier independent explorations, starting from the 4×4 context, attribute exploration expanded it to a 10×4 context, which then served as the input for object exploration. Because this enlarged context was substantially larger than the 4×7 context used in the previous standalone object exploration, the number of errors increased markedly (by approximately 40–50%). Nevertheless, the qualitative nature of the generated meanings remained similar, predominantly aligning with interpretations such as "*distribute among others*" and "*divide a single physical object into constituent parts*". Figure 14 presents the concept lattice generated following the sequential exploration.

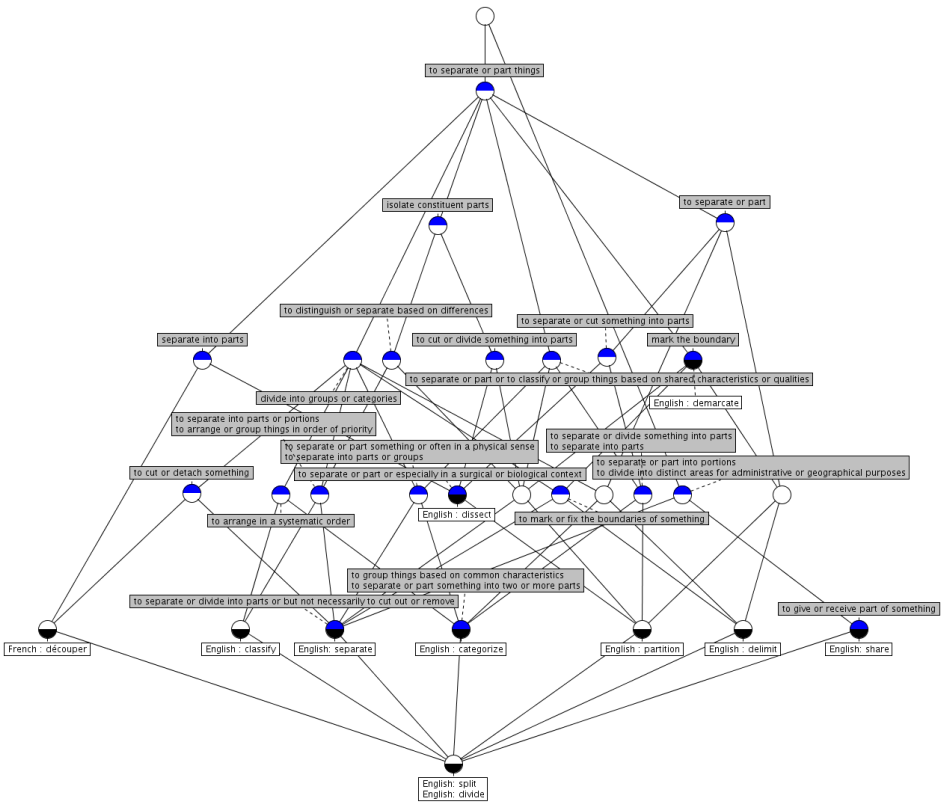
to separate or part things
isolate constituent parts
to separate or part
to distinguish or separate based on differences
to separate or cut something into parts
separate into parts
to cut or divide something into parts
mark the boundary
to separate or part or to classify or group things based on shared characteristics or qualities
divide into groups or categories
English : demarcate
to separate into parts or portions
to arrange or group things in order of priority
to separate or divide something into parts
to separate into parts
to separate or part something or often in a physical sense
to separate into parts or groups
to separate or part into portions
to divide into distinct areas for administrative or geographical purposes
to cut or detach something
to separate or part or especially in a surgical or biological context
English : dissect
to arrange in a systematic order
to mark or fix the boundaries of something
to group things based on common characteristics
to separate or part something into two or more parts
to separate or divide into parts or but not necessarily to cut out or remove
to give or receive part of something
French : découper
English : classify
English: separate
English : categorize
English : partition
English : delimit
English: share
English: split
English: divide

Figure 14. Resulting Concept Lattice After Sequential Attribute and Object Exploration

4.2.4 Conclusion

This section systematically evaluated LLM-supported conceptual exploration in Formal Concept Analysis, examining how different prompting strategies influence the discovery and validation of semantic implications. Structured prompts (S2 and S3) consistently outperformed minimal prompting (S1) in reducing error rates. Notably, S2 emerged as the most effective strategy, confirming more implications proportionally than S3 while maintaining similar error counts (Tables 2a and 2b). This suggests that S3's extensive instructions may overload the model, leading to overly conservative responses, while S2 strikes an optimal balance between guidance and flexibility.

Attribute exploration proved more robust than object exploration, successfully discovering diverse vocabulary across semantic categories. Expanding the context from 4×4 to 4×8 enabled longer exploration and discovery of more nuanced vocabulary. Object exploration required simplification through unit implications to manage cognitive load, yet still exhibited higher error rates and more limited semantic coverage.

Despite improved prompting, several fundamental issues remained. The system exhibited semantic inconsistency, associating the same lexical items with different meanings across runs. Significant redundancy emerged, with multiple paraphrastic variants of identical concepts treated as distinct discoveries (e.g., "to give part of something to someone/somebody/someone else"). The system also demonstrated incomplete semantic coverage, systematically omitting less frequent semantic frames like "distribute among themselves" and "put into containers," likely due to training distribution biases favoring canonical associations

It is also important to note that the experiments reported here utilized the fully automated mode of the system. Our framework also provides manual and assisted modes where human experts can intervene at critical decision points. In these modes, many of the identified limitations such as semantic redundancy, inconsistent counterexamples, and incomplete coverage can be mitigated through expert judgment and guidance. The manual mode allows experts to validate each implication and counterexample before incorporation, while the assisted mode enables selective intervention when the system encounters ambiguous cases or generates suspicious results. These hybrid approaches combine the efficiency of automated exploration with the semantic expertise and critical judgment of human analysts, offering a more robust and reliable pathway for conceptual structure discovery.

Future work should explore semantic similarity detection to reduce redundancy, bias mitigation for improved coverage of rare semantic dimensions, ensemble methods for consistency, systematic evaluation of manual versus assisted versus automated modes, and iterative refinement with expert feedback. The most effective approach may lie not in perfecting automation, but in designing intelligent human-AI collaboration workflows that leverage complementary strengths of both.

References

[1] Andreeva, A.A.: Verbs of Dividing and Sharing: Towards a Typology, Bachelor Thesis, HSE University, Moscow, 2024 (in Russian)

[2] Croft, W.: Radical Construction Grammar: Syntactic Theory in Typological Perspective. Oxford: Oxford University Press (2001)

[3] Croft, W., Poole, K. T.: Inferring Universals from Grammatical variation: Multidimensional Scaling for Typological Analysis. Theoretical Linguistics 34.1 (2008):1–37

[4] Cysouw, M., Haspelmath, M.: Indefinite pronouns (Oxford Studies in Typology and Linguistic Theory). Oxford: Clarendon Press, 1997. pp. xvi+364., Journal of Linguistics 37 (2001) 593–625. doi:10.1017/S0022226701231351

[5] de Haan, F.: Building a semantic map: top-down versus bottom-up approaches. Linguistic Discovery 8 (2010): 102

[6] Freeman, L.C., White D.R.: Using Galois Lattices to Represent Network Data. Sociological Methodology 23 (1993): 127–146

[7] Ganter, B., Obiedkov, S.: Conceptual Exploration. Springer (2016)

[8] Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Berlin/Heidelberg (1999)

[9] Georgakopoulos, T., Grossman, E., Nikolaev, D., Polis, S.: Universal and macro-areal patterns in the lexicon: A case-study in the perception-cognition domain, Linguistic Typology 26 (2022) 439–487. doi:10.1515/lingty-2021-2088

[10] Georgakopoulos, T. & Polis, S.: The semantic map model: State of the art and future avenues for linguistic research, Language and Linguistics Compass 12(2) (2018), e12270. https://doi.org/10.1111/lnc3.12270

[11] Georgakopoulos, T. & Polis, S.: New avenues and challenges in semantic map research (with a case study in the semantic field of emotions). Zeitschrift für Sprachwissenschaft, vol. 41, no. 1 (2022), 1-30. https://doi.org/10.1515/zfs-2021-2039

[12] Großkopf A., Harras G.: Begriffliche Erkundung semantischer Strukturen von Sprechaktverben. In: Stumme G., Wille R. (Eds.) Begriffliche Wissensverarbeitung: Methoden und Anwendungen (1999)

[13] Guigues, J.L., Duquenne, V.: Famille minimale d'implications informatives resultant d'un tableau de données binaires. Mathématiques et Sciences Humaines 24(95) (1986), 5–18

[14] Kipke U., Wille R.: Formale Begriffsanalyse erläutert an einem Wortfeld. LDV–Forum. Vol. 5 (1987)

[15] Priss, U.: Linguistic Applications of Formal Concept Analysis. In Ganter, B., Stumme, G., Wille, R. (eds.): Formal Concept Analysis: Foundations and Applications. Lecture Notes in Computer Science, vol. 3626, Springer (2005): 149–160

[16] Rakhilina, E., Reznikova, T.: A Frame-based Methodology for Lexical Typology. The Lexical Typology of Semantic Shifts, 58 (2016): 95–130

[17] Rakhilina, E., Ryzhova, D., Badryzlova, Y.: Lexical typology and semantic maps: Perspectives and challenges, Zeitschrift für Sprachwissenschaft 41 (2022) 231–262. doi:doi:10.1515/zfs-2021-2046

[18] Ryzhova, D., Obiedkov, S.: Formal concept lattices as semantic maps. In Ekaterina L. Chernyak (ed.), Computational linguistics and language science, 78–87. Aachen CEUR Workshop Proceedings (2017)

[19] Wälchli, B., Cysouw, M.: Lexical Typology through Similarity Semantics: Toward a Semantic Map of Motion Verbs. De Gruyter 50.3 (2012): 671–710

[20] Zwarts, J.: Commentary on de Haan—Can semantic maps be built purely bottom-up? Linguistic Discovery, 8(1) (2010), 123–124. https://doi.org/10.1349/ps1.1537-0852.a.368

Appendix A: Description of the frames

Frame	Description	Example
1. divide a single physical object into constituent parts	A situation when the subject violates the integrity of a single physical object so that the object is separated into parts; however, the subject does not transfer the parts of the object formed as a result of the division to other participants or group them in any way. The tool in this situation is either absent ('divide the dough (with hands) into four parts'), or its use is implied, but is not named in the context ('divide the tablet into four parts').	Once dough has risen, **divide** it into four even pieces.
2. divide a set of elements into groups	A situation where, firstly, the object is a set of elements, secondly, the subject performs the division, distributing the parts of the object into certain groups. Examples: 'divide students into groups', 'divide people into friends and foes', 'divide films into sad and funny'.	**Divide** the children into five groups and give each group one of the puzzles you made.
3. mark the boundary	A situation in which the subject draws one or more conventional boundaries within an object lying on a plane; the integrity of the object is not literally violated. Examples: 'divide the forest into sections', 'divide the drawn circle in half', 'divide the city / room into two parts'.	Beyond the second cluster of towers, an immense rectangle of vegetation **divided** the city into east and west.
4. isolate constituent parts of a non-physical object	This involves breaking down a 'whole' abstract object into some components – for example, 'to divide a novel into chapters'; the integrity of the object, as in the case of delineating boundaries, remains intact. In this situation, the idea of separation into parts is preserved, but the object is non-physical.	**Divide** long surveys into sections or pages.
5. put into containers	The subject separates a whole object or an object consisting of elements, distributing the resulting parts into containers – for example, 'arranges pasta into plates'.	**Divide** the custard among the pastry tart cases and set aside to cool.
6. cause to move or be apart	The subject removes two people from each other, stopping their communication.	You know, the cops catch us, they'd take her away and they'd **split** us **up**, forever.
7. distribute among others	A situation in which parts of a divided object are distributed among a certain set of participants. The subject distributes parts among other people and is not one of the recipients: cf. the contexts 'to divide the acquired property among four grandchildren', 'to divide the duties among the employees'.	The $10,000 cash prize will be **divided** equally among the participants.
8. distribute among themselves	A situation in which parts of a divided object are distributed among a certain set of participants. Several subjects take parts of the object for	The two conspirators then **divided** the money between them.

	themselves: cf. ‘divide a chocolate bar between two’, ‘divide stolen money among one self’s, ‘divide the father’s land after his death’.	

Appendix B: Formats of the input file

The system accepts context input in multiple formats to accommodate various data sources and user preferences:

1. CSV (Comma-Separated Values)
   - Standard tabular format with objects as rows and attributes as columns
   - Headers indicate attribute names
   - Binary values (X/., 0/1, True/False, Yes/No) indicate attribute presence
2. CXT (Burmeister Format)
   - Native formal context format used in FCA tools
   - Structured text format containing objects, attributes, and their incidence relation
4. Excel/XLSX
   - Spreadsheet format for convenient manual editing
   - Similar structure to CSV with enhanced formatting options

Each format is automatically detected and parsed by the system upon upload.

Appendix C: Prompts used for Explorations

S1:

Current Word Meaning List:
1. "separate into parts"
2. "divide into groups or categories"
3. "mark the boundary"
4. "isolate constituent parts"

Check the below hypothesis :
Every verb that conveys the meaning(s) "isolate constituent parts" also conveys the meaning(s) "mark the boundary" and "divide into groups or categories"?

Return the result as a valid JSON object using the following logic :

If the hypothesis holds
{
"output": "YES"
}

Otherwise, return only one counter example refuting the hypothesis
{
"output": "NO",
"verb": "<Language of the verb> : <name of the verb>”,
"meaning": ["<list of all meanings associated to the verb>"]
}.

S2 :

We are analyzing word meanings for multilingual verbs using the framework of Formal Concept Analysis (FCA).

Current Word Meaning List:
1. "separate into parts"
2. "divide into groups or categories"
3. "mark the boundary"
4. "isolate constituent parts"

Hypothesis to Test:
Every verb that conveys the meaning(s) "isolate constituent parts" also conveys the meaning(s) "divide into groups or categories" and "mark the boundary"?

Instructions:
1. Search for all verbs that include the meaning(s) "isolate constituent parts".
3. For each remaining verb:
   - Check whether it also conveys the meanings "divide into groups or categories" and "mark the boundary"
4. Return the result as a valid JSON object using the following logic :

If the hypothesis holds (i.e., all relevant verbs have "divide into groups or categories" and "mark the boundary")
{
"output": "YES"
}

Otherwise, return a verb that has all of the following meaning(s): "isolate constituent parts", but does not have at least one of the following meaning(s): "divide into groups or categories" or "mark the boundary"
{
"output": "NO",
"verb": " <Language of the verb> : <name of the verb> ",
"meaning": ["isolate constituent parts"," "<other meanings from the Word Meaning List, if any apply>"], }

Respond with only a valid JSON object. Do not include markdown syntax (like triple backticks) or any explanatory text.

S3:

We are analyzing word meanings for multilingual verbs using the framework of Formal Concept Analysis (FCA).

Word Meaning List:
0. "separate into parts" (e.g. Once dough has risen, divide it into four even pieces.)
1. "divide into groups or categories" (e.g. Divide the children into five groups and give each group one of the puzzles
you made.)
2. "mark the boundary" (e.g. Beyond the second cluster of towers, an immense rectangle of vegetation

divided the city into east and west.)
3. "isolate constituent parts" (e.g. Divide long surveys into sections or pages.)

Already Checked Verbs:
Russian: delit', Russian: delit'sya, Russian: razbivat', Russian: razluchat'

Hypothesis to Test:
Every verb that conveys the meaning(s) "isolate constituent parts" also conveys the meaning(s) "mark the boundary" and "divide into groups or categories"?

Instructions:
1. Search for all verbs that include the meaning(s) "isolate constituent parts".
2. Ignore any verbs already in the checked list.
3. For each remaining verb:
    - Check whether it also conveys the meanings "mark the boundary" and "divide into groups or categories"
4. Return the result as a valid JSON object using the following logic :

If the hypothesis holds (i.e., all relevant verbs have "mark the boundary" and "divide into groups or categories")
{
"output": "YES"
}

Otherwise, return a verb that has all of the following meaning(s): "isolate constituent parts", but does not have at least one of the following meaning(s): "mark the boundary" or "divide into groups or categories"
{
"output": "NO",
"verb": " <Language of the verb> : <name of the verb> ",
"meaning": ["isolate constituent parts"," "<other meanings from the Word Meaning List, if any apply>"],
"language": " <Give the language of the verb>"
"explanation": " <Give a brief explanation of your result, and also describe the general meaning of the verb and also give from which language the verb is taken>"
"example": " <Explain your results using some examples for all the meanings, in the same language of the verb>"}"

Constraints:
- Ensure the returned verb is not in the already checked list.
- Use all the meanings from the list that applies to this word.
- Do not include meanings not on the list.

Respond with only a valid JSON object. Do not include markdown syntax (like triple backticks) or any explanatory text.